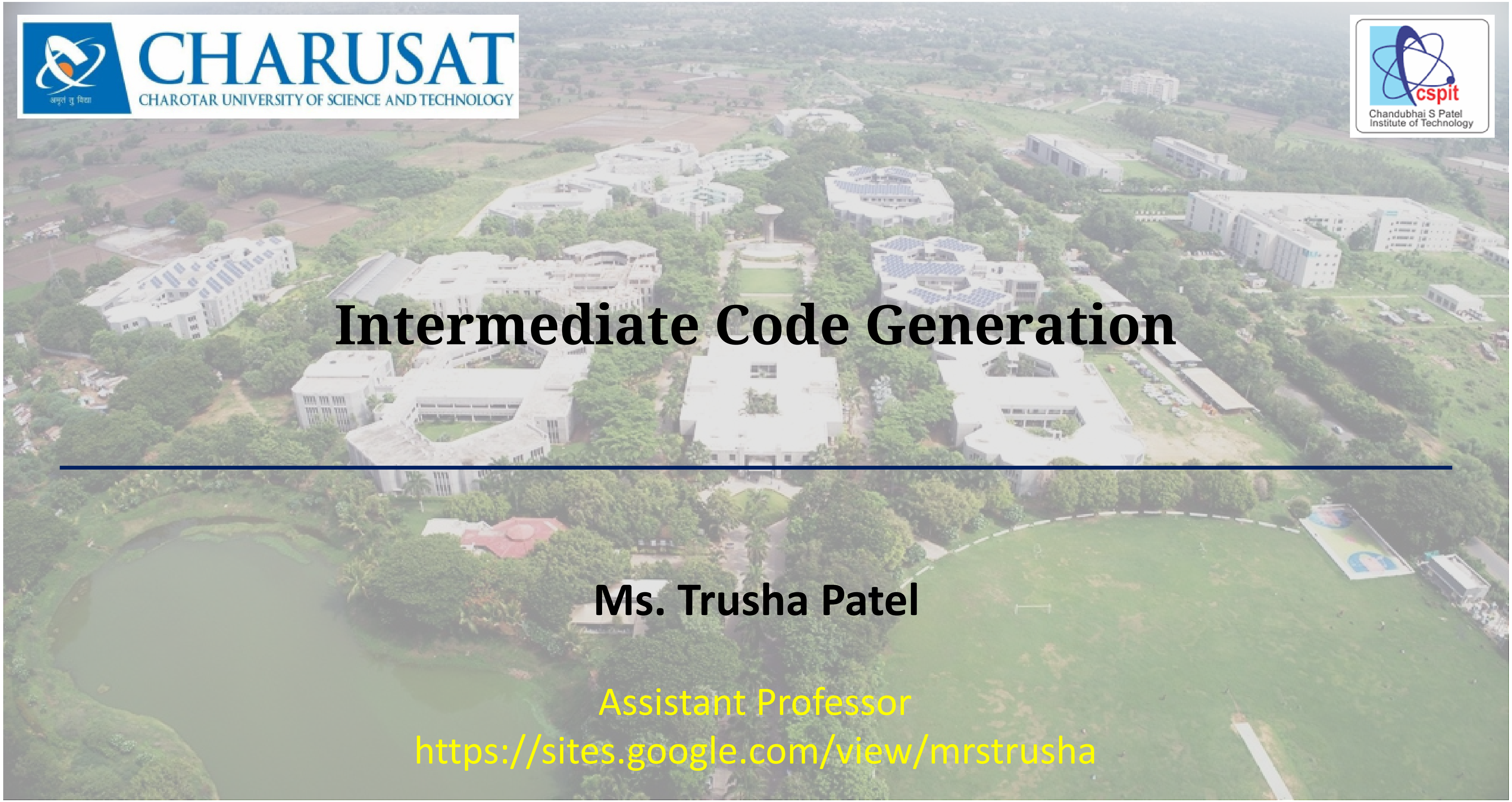
CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

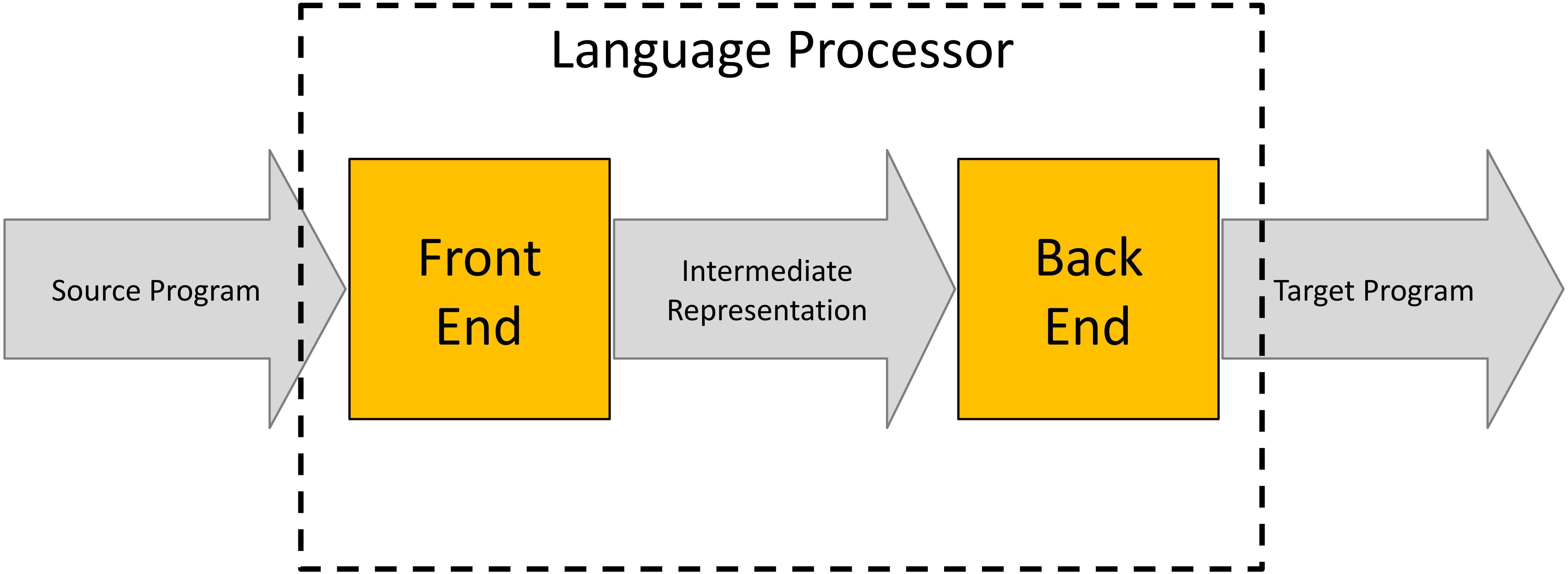
cspit
Chandubhai S Patel Institute of Technology



# Intermediate Code Generation

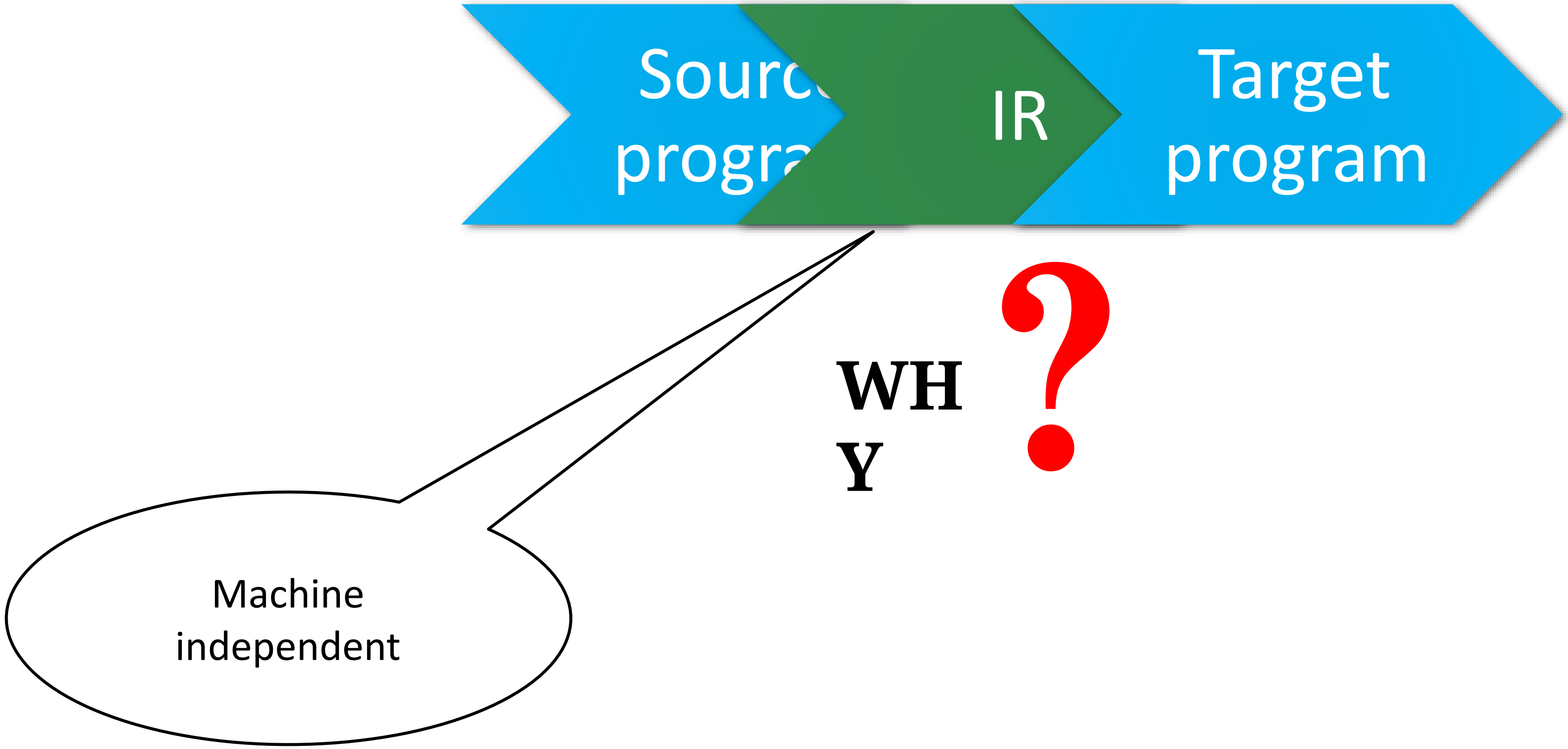
**Ms. Trusha Patel**

Assistant Professor
https://sites.google.com/view/mrstrusha


Language Processor
Source Program
Front End
Intermediate Representation
Back End
Target Program

# IR

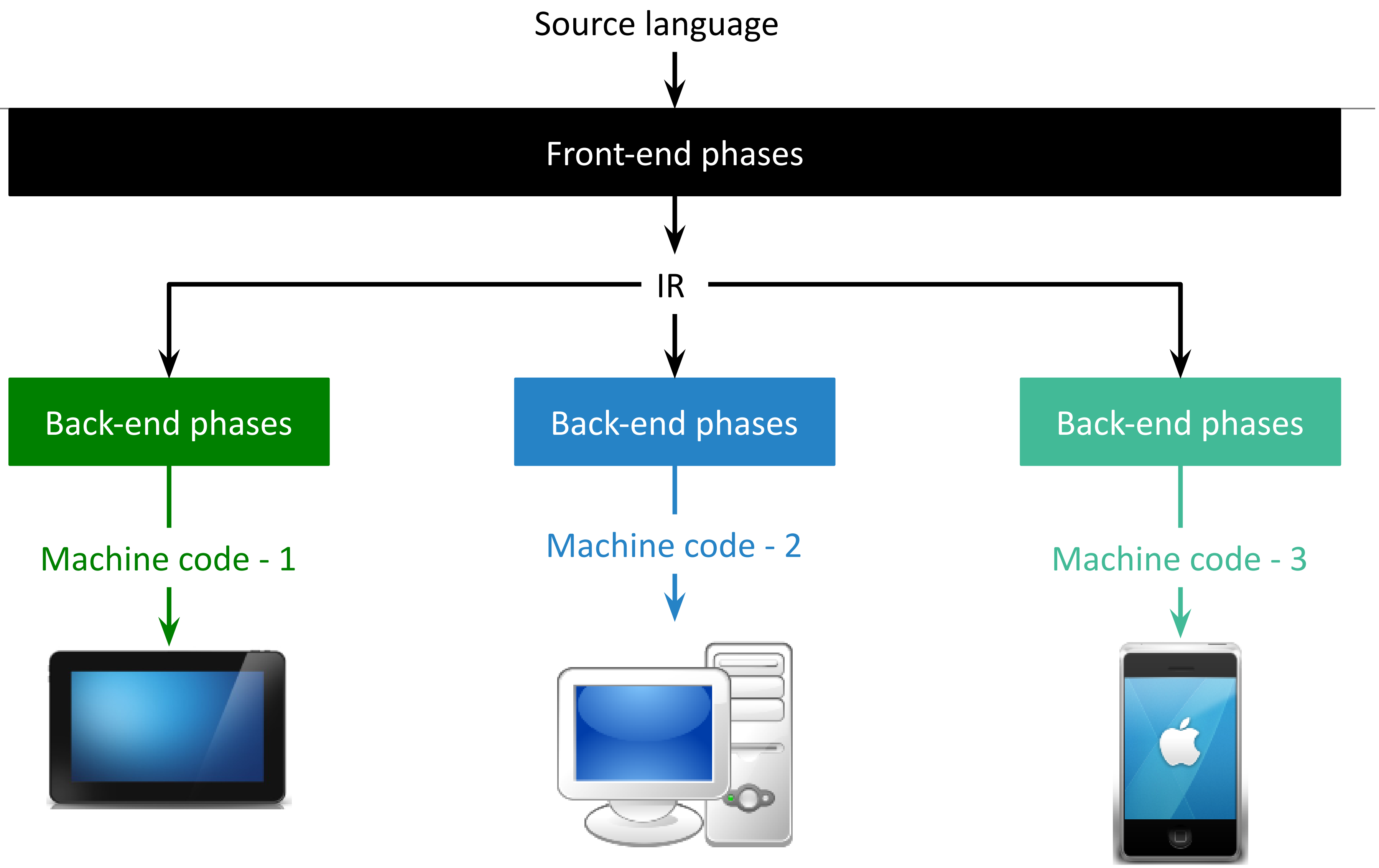
INTERMEDIATE REPRESENTATION



**WH Y ?**

# Advantages

1



Source language

Front-end phases

IR

Back-end phases

Back-end phases

Back-end phases

Machine code - 1

Machine code - 2

Machine code - 3

# Advantages

2

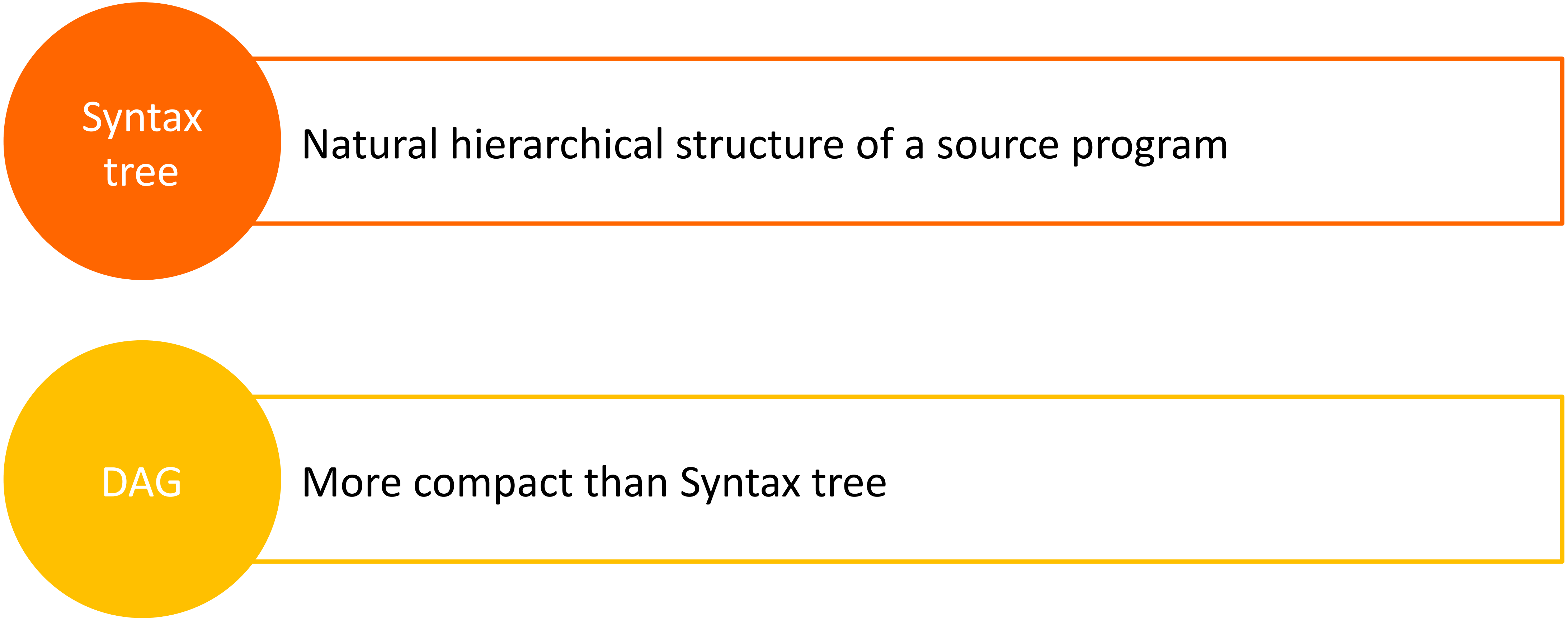
Can apply machine independent code optimizer on IR

# Intermediate Language

Syntax tree

Postfix notation

Three address code



Syntax tree

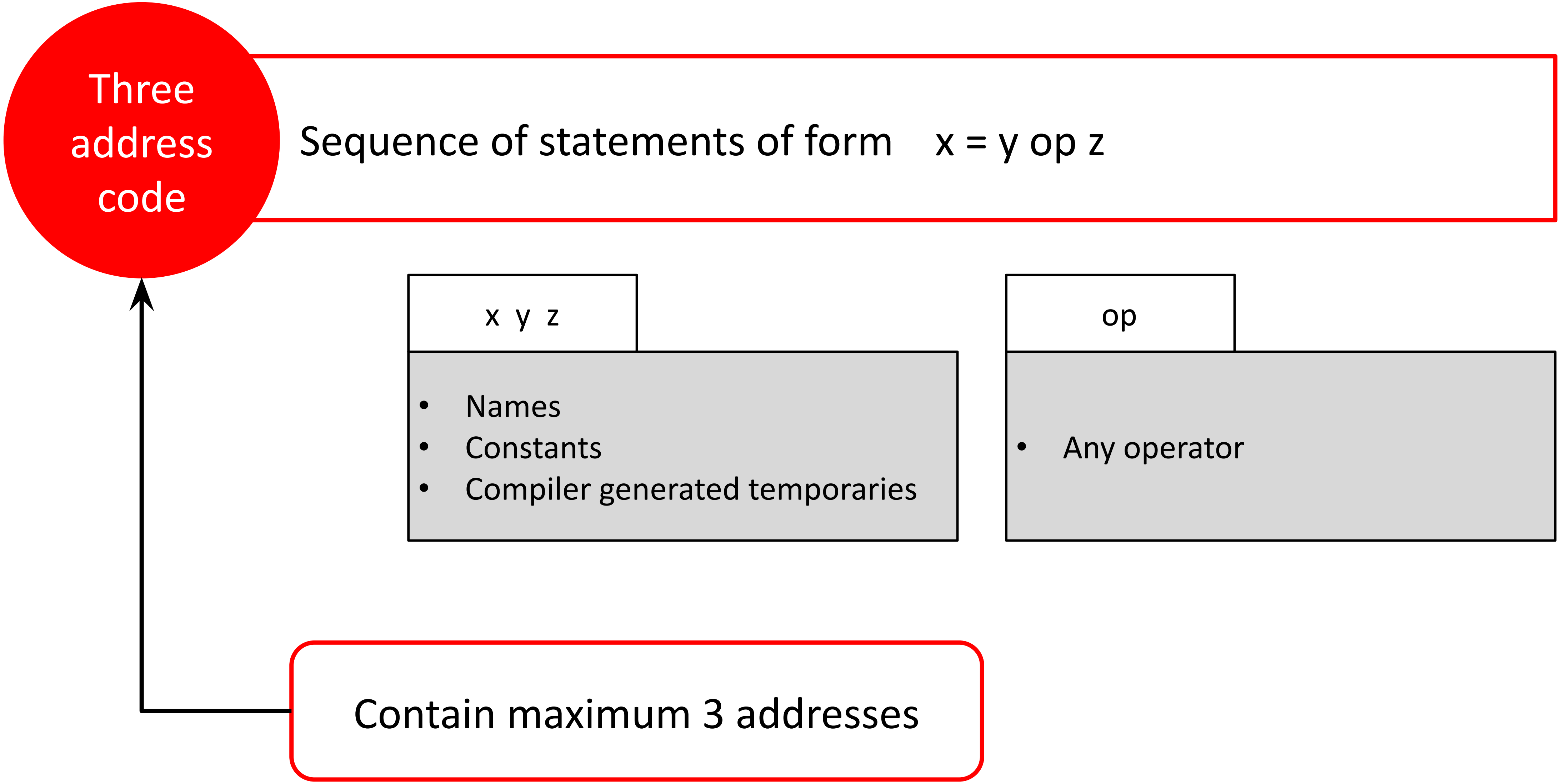
Natural hierarchical structure of a source program

DAG

More compact than Syntax tree

Postfix notation

Linearized representation of syntax tree



Three address code

Sequence of statements of form x = y op z

x y z

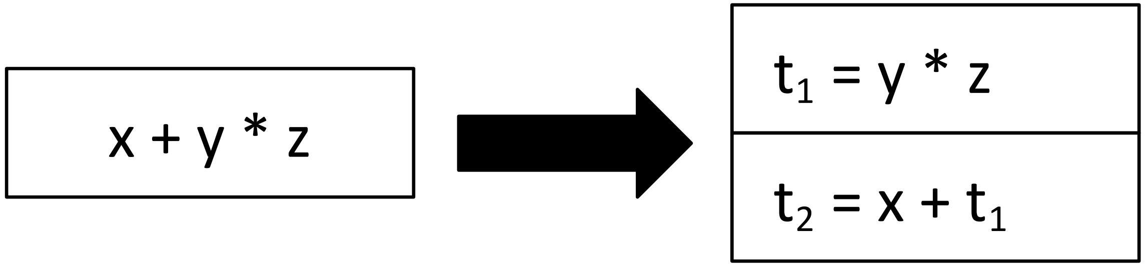
- Names
- Constants
- Compiler generated temporaries

op

- Any operator

Contain maximum 3 addresses

# Three address code



$x + y * z$

$t_1 = y * z$

$t_2 = x + t_1$

Source language expression

Three Address Code

# Three address code

$t_1 = - c$

$t_2 = b * t_1$

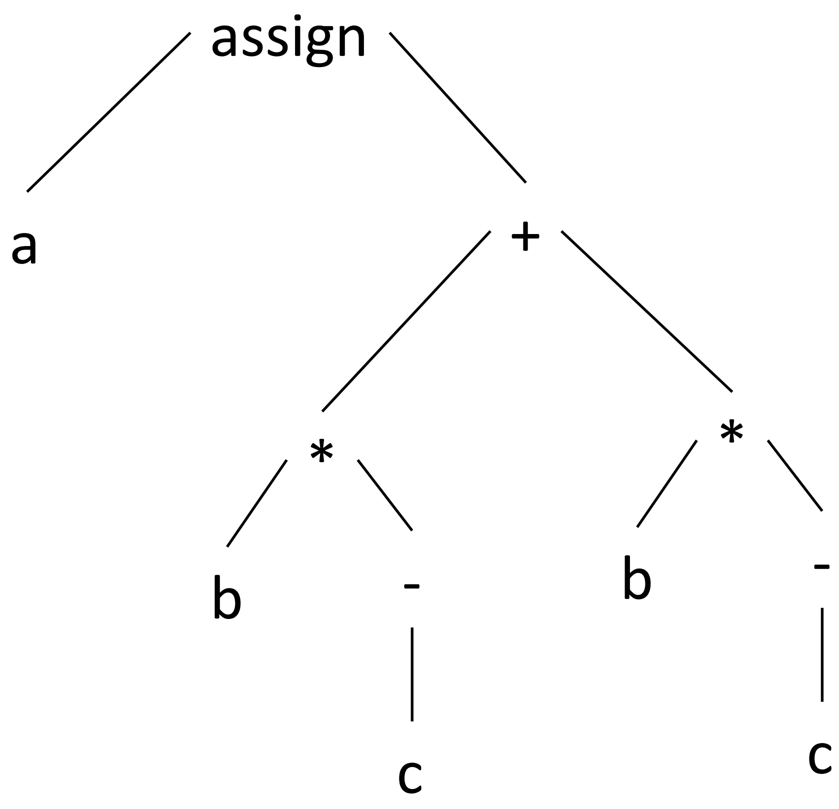
$t_3 = - c$

$t_4 = b * t_3$

$t_5 = t_2 + t_4$

$a = t_5$

**Three Address Code**



**Syntax Tree**

Three address code

$t_1 = - c$

$t_2 = b * t_1$

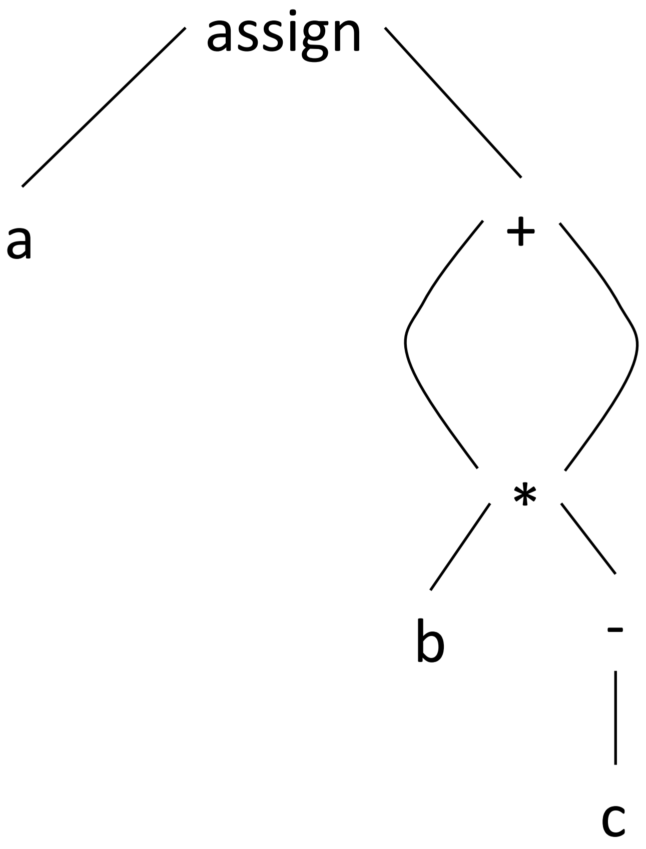
$t_3 = - c$

$t_4 = b * t_3$

$t_5 = t_2 + t_4$

$a = t_5$

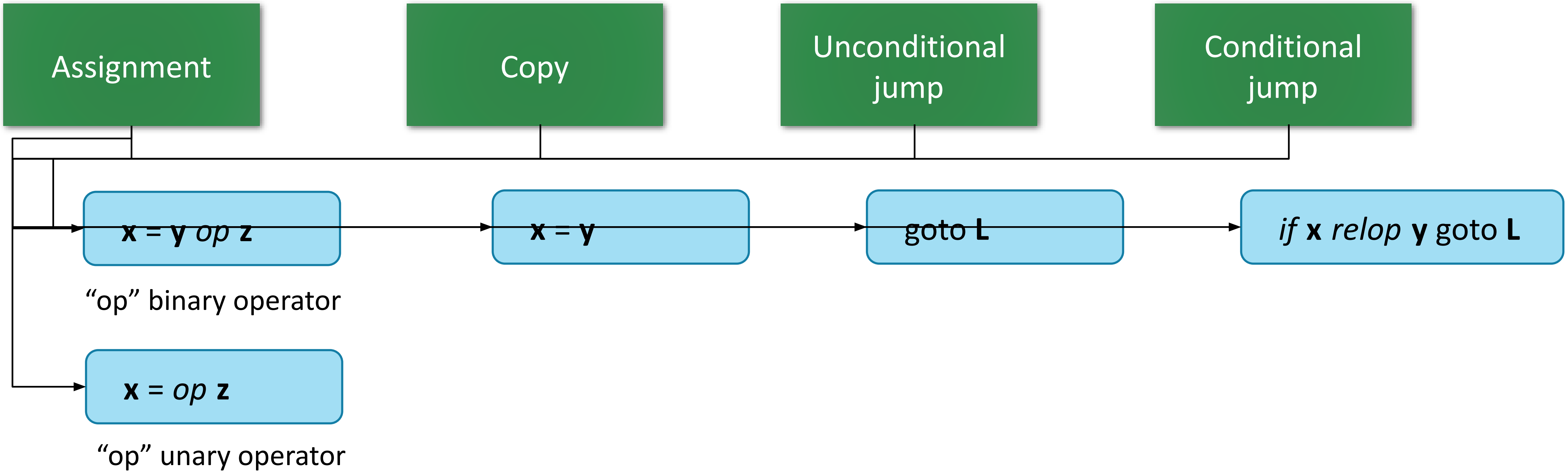
**Three Address Code**



**DAG**

# Three address code

## Types of statements



Assignment

**x** = **y** *op* **z**

“op” binary operator

**x** = *op* **z**

“op” unary operator

Copy

**x** = **y**

Unconditional jump

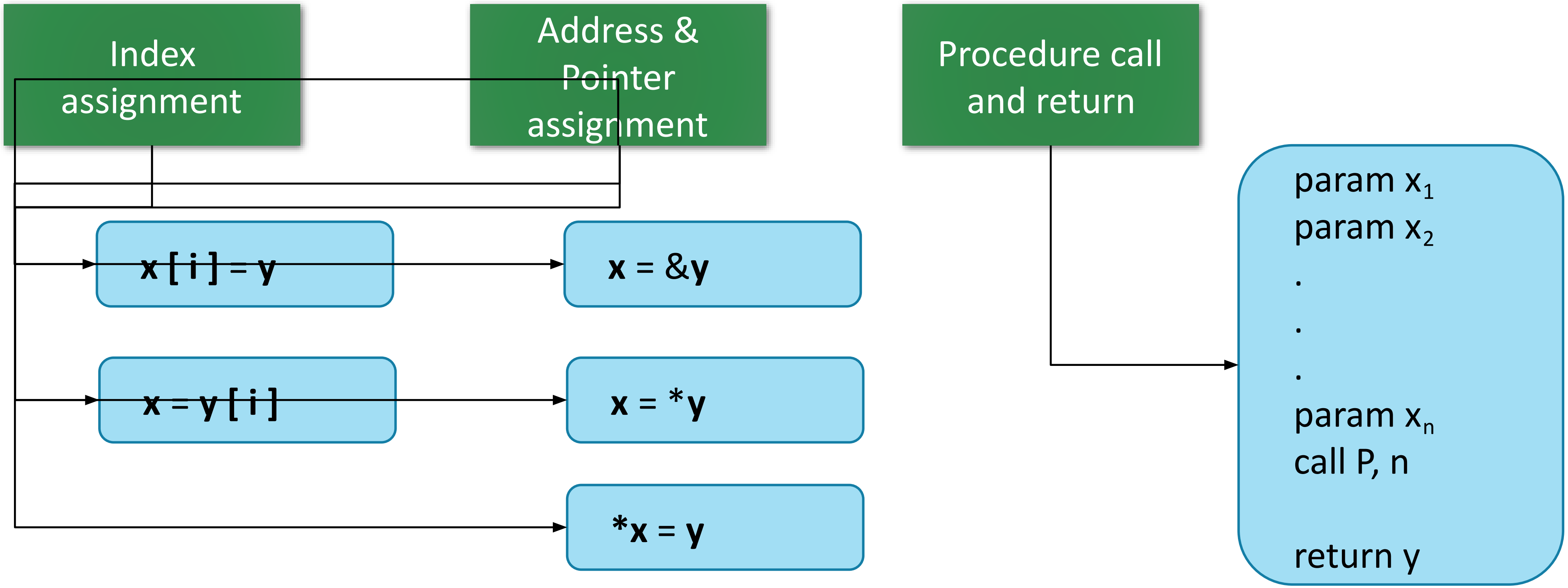
goto **L**

Conditional jump

*if* **x** *relop* **y** goto **L**

Three address code

# Types of statements



Index assignment

Address & Pointer assignment

Procedure call and return

x [ i ] = y

x = y [ i ]

x = &y

x = *y

*x = y

param $x_1$
param $x_2$
.
.
.
param $x_n$
call P, n

return y

Three address code

# Implementation

Quadruples

Triples

Indirect Triples

Three address code

# Implementation

Quadruples

Triples

Indirect Triples

Record structure of 4 fields

| | op | arg1 | arg2 | result |
|---|---|---|---|---|
| (0) | - | c | | $t_1$ |
| (1) | * | b | $t_1$ | $t_2$ |
| (2) | - | c | | $t_3$ |
| (4) | * | b | $t_3$ | $t_4$ |
| (5) | + | $t_2$ | $t_4$ | $t_5$ |
| (6) | = | $t_5$ | | a |

| |
|---|
| $t_1$ = - c |
| $t_2$ = b * $t_1$ |
| $t_3$ = - c |
| $t_4$ = b * $t_3$ |
| $t_5$ = $t_2$ + $t_4$ |
| a = $t_5$ |

Three address code

# Implementation

Quadruples

Triples

Indirect Triples

Record structure of 3 fields

Avoid to enter temporary in table, Refer temporary with position

| | op | arg1 | arg2 |
|---|---|---|---|
| (0) | - | c | |
| (1) | * | b | (0) |
| (2) | - | c | |
| (4) | * | b | (2) |
| (5) | + | $t_2$ | (3) |
| (6) | = | $t_5$ | (5) |

| |
|---|
| $t_1$ = - c |
| $t_2$ = b * $t_1$ |
| $t_3$ = - c |
| $t_4$ = b * $t_3$ |
| $t_5$ = $t_2$ + $t_4$ |
| a = $t_5$ |

Three address code

# Implementation

Quadruples

Triples

Indirect Triples

Listing pointers to triples,
rather than listing triples themselves

| | op |
|---|---|
| (0) | (14) |
| (1) | (15) |
| (2) | (16) |
| (4) | (17) |
| (5) | (18) |
| (6) | (19) |

| | op | arg1 | arg2 |
|---|---|---|---|
| (14) | - | c | |
| (15) | * | b | (14) |
| (16) | - | c | |
| (17) | * | b | (16) |
| (18) | + | $t_2$ | (17) |
| (19) | = | $t_5$ | (18) |

# Reference

- Alfred Aho, Ravi Sethi, Jeffrey D Ullman, *Compilers Principles, Techniques and Tools*, Pearson Education Asia.